A Data and Model Visualization System for Android Malware Detection

CPSC 547
Michael Cao
Gabby Xiong
Android Malware Detection

- Cybercriminals target mobile due to large user base
- Rely on machine learning
Android Malware Detection

- Cybercriminals target mobile due to large user base
- Rely on machine learning

Training Samples

- Benign
- Malware
Android Malware Detection

- Cybercriminals target mobile due to large user base
- Rely on machine learning
Android Malware Detection

- Cybercriminals target mobile due to large user base
- Rely on machine learning
Issue with Android Malware Detection

- Focus on producing models with high accuracy
  - What about model attackability?
Issue with Android Malware Detection

- Focus on producing models with high accuracy
  - What about model attackability?

Attackability of the Model is also Important!
Issue with Android Malware Detection

- Focus on producing models with high accuracy
  - What about model attackability?
- Mimicry Attack:
  - Inject features they think represent benign to mislead detection
Issue with Android Malware Detection

- Different sets of benign samples, different impacts on results
Issue with Android Malware Detection

- Different sets of benign samples, different impacts on results

![Diagram showing classification of benign and malware samples with higher detection rate and higher attackability.]
Issue with Android Malware Detection

- Different sets of benign samples, different impacts on results
Issue with Android Malware Detection

- Different sets of benign samples, different impacts on results

How do the benign samples used influence the model?

Exploration requires a lot of trial and errors, and doing so manually would be laborsome!
Issue with Android Malware Detection

- Different sets of benign samples, different impacts on results

Our solution: visualization system to facilitate exploration!
High Level Overview

Input:
- Benign
- Malware
- Training
- Testing Data

Visualization System:
- Sample Distribution
- Feature Distribution
- Train Test
- Model Performance
- Responsive Interaction
- Select Sample
- Select Feature
- Visualization Feedbacks

Output:
- Sample Selected
- Model Result
High Level Overview

**Input**
- Benign
- Malware
- Training
- Testing Data

**Visualization System**
- Sample Distribution
- Feature Distribution
- Train Test
- Model Performance
- Responsive Interaction
  - Select Sample
  - Select Feature
  - Visualization Feedbacks

**Output**
- Sample Selected
- Model Result

**Researcher**
High Level Overview

Input
- Benign
- Malware
- Training
- Testing Data

Visualization System
- Sample Distribution
- Feature Distribution
- Responsive Interaction
- Train Test
- Model Performance

Output
- Sample Selected
- Model Result

Select Sample
Select Feature
Visualization Feedbacks

Researcher
High Level Overview

**Input**
- Benign
- Malware
- Training
- Testing Data

**Visualization System**
- Sample Distribution
- Feature Distribution
- Train Test
- Model Performance
- Responsive Interaction

**Output**
- Sample Selected
- Model Result

- Select Sample
- Select Feature
- Visualization Feedbacks

Researcher
High Level Overview

Input
- Benign
- Malware
- Training
- Testing Data

Visualization System
- Sample Distribution
- Feature Distribution
- Train Test
- Model Performance
- Responsive Interaction
- Select Sample
- Select Feature
- Visualization Feedbacks

Output
- Sample Selected
- Model Result

Researcher
DREBIN: Case Study

- Well-known Android malware detection technique [1]
- Eight categories of features
DREBIN: Case Study

- Well-known Android malware detection technique [1]
- Eight categories of features

![Diagram showing Android Manifest File with App Components, Intent Filter, Requested Permission, and Hardware. The diagram also shows Benign and Malware categories.]
DREBIN: Case Study

- Well-known Android malware detection technique [1]
- Eight categories of features

![Diagram showing the process of DREBIN analysis]

- Android Manifest File
  - App Components
  - Intent Filter
  - Requested Permission
  - Hardware
- Application Bytecode
  - Used Permission
  - Suspicious API calls
  - Restricted API calls
  - URL Domains
DREBIN: Case Study

- Binary values indicate presence / absence of feature
- Concatenate features from all samples to form feature space
- SVM classifier
Data

Android Sample Data:
- 5,000 malware from VirusTotal
- 10,000 benign from Google Play
- From year 2011 to 2019

Types of Data:
- Android Samples
  - Temporal
  - Drebin Features
- Model
  - Performance
  - Attackability
Tasks (Training)

- Exploring training data distribution
  - Compare benign and malware distributions
    - View similarity between samples
      - What contributes to the similarity/dissimilarity
        - Individual feature
        - Feature category
      - Observe similarity/dissimilarity by including/discluding features
Tasks (Training)

- Exploring training data distribution
  - Compare benign and malware distributions
    - View similarity between samples
Tasks (Training)

- Exploring training data distribution
  - Compare benign and malware distributions
    - View similarity between samples
  - What contribute to the similarity / dissimilarity
    - Individual feature
    - Feature category
Tasks (Training)

- Exploring training data distribution
  - Compare benign and malware distributions
    - View similarity between samples
  - What contribute to the similarity / dissimilarity
    - Individual feature
    - Feature category
  - Observe similarity/dissimilarity by including/discluding features
Tasks (Training)

- Exploring training data distribution
  - Compare benign and malware distributions
    - View similarity between samples
  - What contribute to the similarity / dissimilarity
    - Individual feature
    - Feature category
  - Observe similarity/dissimilarity by including/discluding features

- Select particular set of samples and features to train model
Tasks (Testing)

- Exploring the model
  - Investigate model performance and attackability
    - Accuracy, # features to flip detection
Tasks (Testing)

● Exploring the model
  ○ Investigate model performance and attackability
    ■ Accuracy, # features to flip detection
  ○ Understand important features learned by model
    ■ Weights of features assigned by the model
Tasks (Testing)

- Exploring the model
  - Investigate model performance and attackability
    - Accuracy, # features to flip detection
  - Understand important features learned by model
    - Weights of features assigned by the model
  - Compare training and testing sample distributions
    - Locate misclassified samples
Tasks (Testing)

- Exploring the model
  - Investigate model performance and attackability
    - Accuracy, # features to flip detection
  - Understand important features learned by model
    - Weights of features assigned by the model
  - Compare training and testing sample distributions
    - Locate misclassified samples
  - Interpret why certain samples being misclassified
Demo

Working scenario of the tool
Limitations

- Limited to analyze “Drebin” Android Malware Detection Tool
Limitations

- Limited to analyze “Drebin” Android Malware Detection Tool

- Limited functionality
  - No way to tune dimensional reduction results
Limitations

● Limited to analyze “Drebin” Android Malware Detection Tool

● Limited functionality
  ○ No way to tune dimensional reduction results

● Dealing with large feature space
  ○ Scalability
  ○ More flexible approaches to select features
  ○ User has no idea on features for particular samples
Limitations

- No mechanism for cross experiment comparisons
  - Add juxtaposed view for comparisons
  - Less control over the testing samples

- Few “What-If” functionalities included in the system
  - Allow user to modify / oversample training samples
Lesson Learned

● Prototype, Prototype, Prototype!
Lesson Learned

- Prototype, Prototype, Prototype!
Thank you! Q&A
Intro & Framing

● Research Topic (a tool to facilitate exploring the relationship between training data and resulting model)?
  ○ How to select the set of benign samples that results in the best performance?
  ○ OR
  ○ Current process of performing such exploratory tasks are time consuming?